H 5

```
   SSSSSSSSSSSS     000000000     RRRRRRRRRRRR     TTTTTTTTTTTTTTT     333333333        222222222
   SSSSSSSSSSSS     000000000     RRRRRRRRRRRR     TTTTTTTTTTTTTTT     333333333        222222222
   SSSSSSSSSSSS     000000000     RRRRRRRRRRRR     TTTTTTTTTTTTTTT     333333333        222222222
SSS              000         000  RRR         RRR        TTT        333         333  222         222
SSS              000         000  RRR         RRR        TTT        333         333  222         222
SSS              000         000  RRR         RRR        TTT        333         333  222         222
SSS              000         000  RRR         RRR        TTT                    333              222
SSS              000         000  RRR         RRR        TTT                    333              222
SSS              000         000  RRR         RRR        TTT                    333              222
   SSSSSSSSS     000         000  RRRRRRRRRRRR           TTT                 333              222
   SSSSSSSSS     000         000  RRRRRRRRRRRR           TTT                 333              222
   SSSSSSSSS     000         000  RRRRRRRRRRRR           TTT                 333              222
            SSS  000         000  RRR   RRR              TTT                    333        222
            SSS  000         000  RRR   RRR              TTT                    333        222
            SSS  000         000  RRR   RRR              TTT                    333        222
            SSS  000         000  RRR      RRR           TTT        333         333     222
            SSS  000         000  RRR      RRR           TTT        333         333     222
            SSS  000         000  RRR      RRR           TTT        333         333     222
SSSSSSSSSSSS        000000000     RRR         RRR        TTT           333333333     222222222222222
SSSSSSSSSSSS        000000000     RRR         RRR        TTT           333333333     222222222222222
SSSSSSSSSSSS        000000000     RRR         RRR        TTT           333333333     222222222222222
```

_$2

Pse

SOR

SOR

SOR

SOR

_LI

```
**FILE**ID**CDDMAC                                   K 14



  CCCCCCCC  DDDDDDDD    DDDDDDDD    MM      MM    AAAAAA      CCCCCCCC
  CCCCCCCC  DDDDDDDD    DDDDDDDD    MM      MM    AAAAAA      CCCCCCCC
CC          DD      DD  DD      DD  MMMM  MMMM  AA      AA  CC
CC          DD      DD  DD      DD  MMMM  MMMM  AA      AA  CC
CC          DD      DD  DD      DD  MM  MM  MM  AA      AA  CC
CC          DD      DD  DD      DD  MM  MM  MM  AA      AA  CC
CC          DD      DD  DD      DD  MM      MM  AA      AA  CC
CC          DD      DD  DD      DD  MM      MM  AA      AA  CC
CC          DD      DD  DD      DD  MM      MM  AAAAAAAAAA  CC
CC          DD      DD  DD      DD  MM      MM  AAAAAAAAAA  CC
CC          DD      DD  DD      DD  MM      MM  AA      AA  CC              ....
CC          DD      DD  DD      DD  MM      MM  AA      AA  CC              ....
  CCCCCCCC  DDDDDDDD    DDDDDDDD    MM      MM  AA      AA    CCCCCCCC      ....
  CCCCCCCC  DDDDDDDD    DDDDDDDD    MM      MM  AA      AA    CCCCCCCC      ....

RRRRRRRR      333333      222222
RRRRRRRR      333333      222222
RR      RR  33      33  22      22
RR      RR  33      33  22      22
RR      RR          33          22
RR      RR          33          22
RRRRRRRR          33          22
RRRRRRRR          33          22
RR  RR              33      22
RR  RR              33      22
RR    RR    33      33  22
RR    RR    33      33  22
RR      RR    333333    2222222222
RR      RR    333333    2222222222
```

```
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

!
! Version:      'V04-000'
!
!  TITLE:  CDDMAC                       CDD Macro Require File
!
!  FACILITY:  Common Data Dictionary
!
!  ABSTRACT:
!       This require file contains the CDD Macros used to access the CDD User
!       Interface.
!
!  ENVIRONMENT:
!
!  AUTHOR:  Jeff East and Kenneth J. Marchilena,        21-Oct-80
!
!  MODIFIED BY:
!
!       P.D.Gilbert     31-Jul-1981
!               Fixed bug with $CDD$CREATE_HISTORY with no descriptor parameter
!
!--

!+
!
!       $R(parm)
!
!       This macro checks to make certain a required parameter is present.
!
!-
MACRO
    $R(parm) =
        %IF %NULL (parm) %THEN
            %WARN ('Required parameter ',
            %QUOTE %QUOTE %QUOTE %QUOTE %QUOTE %QUOTE parm, ' missing.')
            0
```

CDD

```
        %ELSE
            parm
        %FI
    %;

KEYWORDMACRO

!
!       status.wlc.v = CDD$CLEAR_CELL (context.rlu.r, list.rlu.r,
!                       cell.rwu.v);
!

    $CDD$CLEAR_CELL (context, list, cell) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CLEAR_CELL           : FORTRAN;

            CDD$CLEAR_CELL (%EXPAND $R(context), %EXPAND $R(list),
                %EXPAND $R(cell))
        END
    %,


!
!       status.wlc.v = CDD$CREATE_ACL_ENTRY (context.rlu.r, [path.rt.dx] ,
!               [node.rlu.r] , position.rwu.v, [grant.rlu.v] , [deny.rlu.v] ,
!               [banish.rlu.v] , [password.rt.dx] ,
!               [terminal.rt.dx] , [uic.rt.dx] , [username.rt.dx]);

    $CDD$CREATE_ACL_ENTRY (context, path, node, position, grant=0, deny=0,
    banish=0, password=0, terminal=0, uic=0, username=0) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_ACL_ENTRY    : FORTRAN;

            CDD$CREATE_ACL_ENTRY (%EXPAND $R(context)
                %IF %NULL (path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                        , 0, 0
                    %ELSE
                        , 0, node
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path, 0
                    %ELSE
                        , path, node
                    %FI
                %FI
                , %EXPAND $R(position), grant, deny, banish, password, terminal, uic,
                    username)
        END
    %,


!
!       status.wlc.v = CDD$CREATE_DIR (context.rlu.r, path.rt.dx, [node.rlu.r] ,
!               [protocol.rt.dx] , [options.rlu.v] , [[location.wlu.r]);
!

    $CDD$CREATE_DIR (context, path, node, protocol, options, location) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_DIR              : FORTRAN;

            CDD$CREATE_DIR (%EXPAND $R(context), %EXPAND $R(path)
                %IF %NULL(node) %THEN
                    %IF %NULL(protocol) %THEN
                        %IF %NULL (options) %THEN
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , 0, 0, 0, location)
                            %FI
                        %ELSE
                            , 0, 0, options
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , location)
                            %FI
                        %FI
                    %ELSE
                        , 0, protocol
                        %IF %NULL (options) %THEN
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , 0, location)
                            %FI
                        %ELSE
                            , options
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , location)
                            %FI
                        %FI
                    %FI
                %ELSE
                    , node
                    %IF %NULL(protocol) %THEN
                        %IF %NULL (options) %THEN
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , 0, 0, location)
                            %FI
                        %ELSE
                            , 0, options
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , location)
                            %FI
                        %FI
                    %ELSE
                        , protocol
                        %IF %NULL (options) %THEN
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , 0, location)
                            %FI
                        %ELSE
                            , options
                            %IF %NULL (location) %THEN
                                )
                            %ELSE
                                , location)
                            %FI
                        %FI
                    %FI
                %FI
        END
    %,


!
!       status.wlc.v = CDD$CREATE_ENTITY_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v, location.wlu.r);
!

    $CDD$CREATE_ENTITY_ATT (context, entity, attribute, location) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_ENTITY_ATT    : FORTRAN;

            CDD$CREATE_ENTITY_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(location))
        END
    %,


!
!       status.wlc.v = CDD$CREATE_ENTITY_LIST_ATT (context.rlu.r, entity.rlu.r,
!               attribute.rlu.v, list_size.rwu.v, location.wlu.r);
!

    $CDD$CREATE_ENTITY_LIST_ATT (context, entity, attribute, list_size,
        location) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_ENTITY_LIST_ATT       : FORTRAN;

            CDD$CREATE_ENTITY_LIST_ATT (%EXPAND $R(context),
                %EXPAND $R(entity), %EXPAND $R(attribute),
                %EXPAND $R(list_size), %EXPAND $R(location))
        END
    %,

!
!      status.wlc.v = CDD$CREATE_FORWARD (context.rlu.r, path.rt.dx,
!           [node.rlu.r] , file.rt.dx , [ options.rlu.v] , [location.wlu.r]);
!

    $CDD$CREATE_FORWARD (context, path, node=0, file, options, location) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_FORWARD       : FORTRAN;

            CDD$CREATE_FORWARD (%EXPAND $R(context), %EXPAND $R(path), node,
                %EXPAND $R(file)
                %IF %NULL (options) %THEN
                    %IF %NULL (location) %THEN
                        )
                    %ELSE
                        , 0, location)
                    %FI
                %ELSE
                    , options
                    %IF %NULL (location) %THEN
                        )
                    %ELSE
                        , location)
                    %FI
                %FI
        END
    %,

!
!       status.wlc.v = CDD$CREATE_HISTORY (context.rlu.r, entity.rlu.r,
!                      facility.rt.dx, access.rlu.v,program.rt.dx,
!                      description.rt.dx);
!

    $CDD$CREATE_HISTORY (context, entity, facility, access, program,
        description) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_HISTORY       : FORTRAN;

            CDD$CREATE_HISTORY (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(facility), %EXPAND $R(access)
```

```
                    %IF %NULL (program) %THEN
                        %IF %NULL (description) %THEN
                            )
                        %ELSE
                            , 0, description)
                        %FI
                    %ELSE
                        %IF %NULL (description) %THEN
                            , program)
                        %ELSE
                            , program, description)
                        %FI
                    %FI
        END
    %,



!
!
!       status.wlc.v = CDD$CREATE_NULL_ATT (context.rlu.r, entity.rlu.r,
!                      attribute.rlu.v);
!

    $CDD$CREATE_NULL_ATT (context, entity, attribute) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_NULL_ATT     : FORTRAN;

            CDD$CREATE_NULL_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute))
        END
    %,



!
!
!       status.wlc.v = CDD$CREATE_NUM_ATT (context.rlu.r, entity.rlu.r,
!                      attribute.rlu.v, value.rl.v);
!

    $CDD$CREATE_NUM_ATT (context, entity, attribute, value) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_NUM_ATT      : FORTRAN;

            CDD$CREATE_NUM_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(value))
        END
    %,



!
!
!       status.wlc.v = CDD$CREATE_STRING_ATT (context.rlu.r, entity.rlu.r,
!                      attribute.rlu.v, value.rt.dx , [value_size.rwu.v]);
```

```
!

    $CDD$CREATE_STRING_ATT (context, entity, attribute, value, value_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_STRING_ATT           : FORTRAN;

            CDD$CREATE_STRING_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(value)
                %IF %NULL (value_size) %THEN
                    )
                %ELSE
                    , value_size)
                %FI
        END
    %,



!
!       status.wlc.v = CDD$CREATE_STRING_LIST_ATT (context.rlu.r, entity.rlu.r,
!               attribute.rlu.v, list_size.rwu.v, location.wlu.r);
!

    $CDD$CREATE_STRING_LIST_ATT (context, entity, attribute, list_size,
        location) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_STRING_LIST_ATT      : FORTRAN;

            CDD$CREATE_STRING_LIST_ATT (%EXPAND $R(context),
                %EXPAND $R(entity), %EXPAND $R(attribute),
                %EXPAND $R(list_size), %EXPAND $R(location))
        END
    %,


!
!      status.wlc.v = CDD$CREATE_TERM (context.rlu.r, path.rt.dx, [node.rlu.r],
!            protocol.rt.dx, options.rlu.v, location.wlu.r , [prior.rt.dx]);
!

    $CDD$CREATE_TERM (context, path, node=0, protocol, options=0,
        location, prior) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$CREATE_TERM         : FORTRAN;

            CDD$CREATE_TERM (%EXPAND $R(context), %EXPAND $R(path), node,
                %EXPAND $R(protocol), options, %EXPAND $R(location)
                %IF %NULL (prior) %THEN
                    )
                %ELSE
                    , prior)
                %FI
        END
    %,


!
!       status.wlc.v = CDD$DELETE_ACL_ENTRY (context.rlu.r, [path.rt.dx] ,
!               [node.rlu.r] , position.rwu.v);
!

    $CDD$DELETE_ACL_ENTRY (context, path, node, position) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$DELETE_ACL_ENTRY    : FORTRAN;

            CDD$DELETE_ACL_ENTRY (%EXPAND $R(context)
                %IF %NULL (path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                        , 0, 0
                    %ELSE
                        , 0, node
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path, 0
                    %ELSE
                        , path, node
                    %FI
                %FI
                , %EXPAND $R(position))
        END
    %,


!
!       status.wlc.v = CDD$DELETE_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v);
!

    $CDD$DELETE_ATT (context, entity, attribute) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$DELETE_ATT          : FORTRAN;

            CDD$DELETE_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute))
        END
    %,


!
!      status.wlc.v = CDD$DELETE_NODE (context.rlu.r, [path.rt.dx] ,
!             [node.rlu.r] , [options.rlu.v]);
!
```

```
    $CDD$DELETE_NODE (context, path, node, options) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$DELETE_NODE          : FORTRAN;

            CDD$DELETE_NODE (%EXPAND $R(context)
                %IF %NULL (path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                    %ELSE
                        , 0, node
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path
                    %ELSE
                        , path, node
                    %FI
                %FI
                %IF %NULL (options) %THEN
                    )
                %ELSE
                    , options)
                %FI
        END
    %,

!
!       status.wlc.v = CDD$FILL_STRING_CELL (context.rlu.r, list.rlu.r,
!                      cell.rwu.v, value.rt.dx , [value_size.rwu.v]);
!

    $CDD$FILL_STRING_CELL (context, list, cell, value, value_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$FILL_STRING_CELL    : FORTRAN;

            CDD$FILL_STRING_CELL (%EXPAND $R(context), %EXPAND $R(list),
                %EXPAND $R(cell), %EXPAND $R(value)
                %IF %NULL (value_size) %THEN
                    )
                %ELSE
                    , value_size)
                %FI
        END
    %,

!
!
!      status.wlc.v = CDD$FIND_NODE (context.rlu.r, [path.rt.dx] ,
!            [node.rlu.r] , [location.wlu.r] , [protocol.wt.dx] ,
!             [protocol-size.wwu.r]);
!
```

```
    $CDD$FIND_NODE (context, path, node, location, protocol, protocol_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$FIND_NODE           : FORTRAN;

            CDD$FIND_NODE (%EXPAND $R(context)
                %IF %NULL(path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                        , 0, 0
                    %ELSE
                        , 0, node
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path, 0
                    %ELSE
                        , path, node
                    %FI
                %FI
                %IF %NULL (location) %THEN
                    %IF %NULL (protocol) %THEN
                        %IF %NULL (protocol_size) %THEN
                            )
                        %ELSE
                            %ERROR ('Protocol-size cannot be used without protocol')
                            )
                        %FI
                    %ELSE
                        , 0, protocol
                        %IF %NULL (protocol_size) %THEN
                            )
                        %ELSE
                            , protocol_size)
                        %FI
                    %FI
                %ELSE
                    , location
                    %IF %NULL (protocol) %THEN
                        %IF %NULL (protocol_size) %THEN
                            )
                        %ELSE
                            %ERROR ('Protocol-size cannot be used without protocol')
                            )
                        %FI
                    %ELSE
                        , protocol
                        %IF %NULL (protocol_size) %THEN
                            )
                        %ELSE
                            , protocol_size)
                        %FI
                    %FI
                %FI
        END
    %,

!
!       status.wlc.v = CDD$FORMAT_ACL_ENTRY (context.rlu.r, node.rlu.r,
!               position.rwu.v, string.wt.dx, [string-size.wwu.r]);
!

    $CDD$FORMAT_ACL_ENTRY (context, node, position, string, string_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$FORMAT_ACL_ENTRY    : FORTRAN;

            CDD$FORMAT_ACL_ENTRY (%EXPAND $R(context), %EXPAND $R(node),
                %EXPAND $R(position), %EXPAND $R(string)
                %IF %NULL (string_size) %THEN
                    )
                %ELSE
                    , string_size)
                %FI
        END
    %,

!
!       status.wlc.v = CDD$GET_ACCESS_RIGHTS (context.rlu.r, [path.rt.dx] ,
!               [node.rlu.r] , rights.wlu.r);
!

    $CDD$GET_ACCESS_RIGHTS (context, path, node, rights) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_ACCESS_RIGHTS   : FORTRAN;

            CDD$GET_ACCESS_RIGHTS (%EXPAND $R(context)
                %IF %NULL (path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                        , 0, 0
                    %ELSE
                        , 0, node
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path, 0
                    %ELSE
                        , path, node
                    %FI
                %FI
                , %EXPAND $R(rights))
        END
    %,
!
!       status.wlc.v = CDD$GET_ACL_ENTRY (context.rlu.r,
!               node.rlu.r, position.rwu.v, grant.wlu.r, deny.wlu.r,
!               banish.wlu.r, password.wt.dx,
!               terminal.wt.dx, uic.wt.dx, username.wt.dx);
!
    $CDD$GET_ACL_ENTRY (context, node, position, grant, deny,
    banish, password, terminal, uic, username) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_ACL_ENTRY       : FORTRAN;

            CDD$GET_ACL_ENTRY (%EXPAND $R(context), %EXPAND $R(node),
                %EXPAND $R(position), %EXPAND $R(grant),
                %EXPAND $R(deny), %EXPAND $R(banish),
                %EXPAND $R(password), %EXPAND $R(terminal), %EXPAND $R(uic),
                %EXPAND $R(username))
            END
        %,



!
!       status.wlc.v = CDD$GET_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v, type.wlu.r, [location.wlg.r] ,
!                       [value.wt.dx] , [value_size.wwu.r]);
!

    $CDD$GET_ATT (context, entity, attribute, type, location, value, value_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_ATT             : FORTRAN;

            CDD$GET_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(type)
                %IF %NULL (location) %THEN
                    %IF %NULL (value) %THEN
                        %IF %NULL (value_size) %THEN
                            )
                        %ELSE
                            , 0, 0, value_size)
                        %FI
                    %ELSE
                        %IF %NULL (value_size) %THEN
                            , 0, value)
                        %ELSE
                            ,0, value, value_size)
                        %FI
                    %FI
                %ELSE
                    %IF %NULL (value) %THEN
                        %IF %NULL (value_size) %THEN
                            , location)
                        %ELSE
                        , location, 0, value_size)
                    %FI
                %ELSE
                    %IF %NULL (value_size) %THEN
                        , location, value)
                    %ELSE
                        , location, value, value_size)
                    %FI
                %FI
            %FI
        END
    %,

!
!       status.wlc.v = CDD$GET_ATTS (context.rlu.r, location.rlu.r,
!                       list.ra.v);
!

    $CDD$GET_ATTS (context, location, list) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_ATTS            : FORTRAN;

            CDD$GET_ATTS (%EXPAND $R(context), %EXPAND $R(location),
                %EXPAND $R(list))
        END
    %,

!
!       status.wlc.v = CDD$GET_ENTITY_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v, location.wlu.r);
!

    $CDD$GET_ENTITY_ATT (context, entity, attribute, location) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_ENTITY_ATT      : FORTRAN;

            CDD$GET_ENTITY_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(location))
        END
    %,

!
!       status.wlc.v = CDD$GET_ENTITY_CELL (context.rlu.r, list.rlu.r,
!                       cell.rwu.v, location.wlu.r);
!

    $CDD$GET_ENTITY_CELL (context, list, cell, location) =
        BEGIN
```

```
            EXTERNAL ROUTINE
                CDD$GET_ENTITY_CELL      : FORTRAN;

            CDD$GET_ENTITY_CELL (%EXPAND $R(context), %EXPAND $R(list),
                %EXPAND $R(cell), %EXPAND $R(location))
        END
    %,


!
!       status.wlc.v = CDD$GET_ENTITY_LIST_ATT (context.rlu.r, entity.rlu.r,
!                      attribute.rlu.v, location.wlu.r ,
!                      [list_size.wwu.r]);
!

    $CDD$GET_ENTITY_LIST_ATT (context, entity, attribute, location,
        list_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_ENTITY_LIST_ATT : FORTRAN;

            CDD$GET_ENTITY_LIST_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(location)
                %IF %NULL (list_size) %THEN
                    )
                %ELSE
                    , list_size)
                %FI
        END
    %,


!
!       status.wlc.v = CDD$GET_NEXT_ATT (context.rlu.r, entity.rlu.r,
!                      attribute.wlu.r, type.wlu.r, [location.wlg.r],
!                      [string.rt.dx] , [value_size.rwu.r]);
!

    $CDD$GET_NEXT_ATT (context, entity, attribute, type, location, string,
                      value_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_NEXT_ATT         : FORTRAN;

            CDD$GET_NEXT_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(type)
                %IF %NULL (location) %THEN
                    %IF %NULL (string) %THEN
                        %IF %NULL (value_size) %THEN
                            )
                        %ELSE
                            , 0, 0, value_size)
                        %FI
                    %ELSE
                    %IF %NULL (value_size) %THEN
                        , 0, string)
                    %ELSE
                        ,0, string, value_size)
                    %FI
                %FI
            %ELSE
                %IF %NULL (string) %THEN
                    %IF %NULL (value_size) %THEN
                        , location)
                    %ELSE
                        , location, 0, value_size)
                    %FI
                %ELSE
                    %IF %NULL (value_size) %THEN
                        , location, string)
                    %ELSE
                        , location, string, value_size)
                    %FI
                %FI
            %FI
    END
%,


!
!       status.wlc.v = CDD$GET_NULL_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v);
!

    $CDD$GET_NULL_ATT (context, entity, attribute) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_NULL_ATT          : FORTRAN;

            CDD$GET_NULL_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute))
        END
    %,


!
!       status.wlc.v = CDD$GET_NUM_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v, value.wl.r);
!

    $CDD$GET_NUM_ATT (context, entity, attribute, value) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_NUM_ATT           : FORTRAN;

            CDD$GET_NUM_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(value))
        END
    %,



!
!       status.wlc.v = CDD$GET_STRING_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v, value.wt.dx , [value_size.wwu.r]);
!

    $CDD$GET_STRING_ATT (context, entity, attribute, value, value_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_STRING_ATT      : FORTRAN;

            CDD$GET_STRING_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(value)
                %IF %NULL (value_size) %THEN
                    )
                %ELSE
                    , value_size)
                %FI
        END
    %,



!
!       status.wlc.v = CDD$GET_STRING_CELL (context.rlu.r, list.rlu.r,
!                       cell.rwu.v, value.wt.dx , [value_size.wwu.r]);
!

    $CDD$GET_STRING_CELL (context, list, cell, value, value_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_STRING_CELL     : FORTRAN;

            CDD$GET_STRING_CELL (%EXPAND $R(context), %EXPAND $R(list),
                %EXPAND $R(cell), %EXPAND $R(value)
                %IF %NULL (value_size) %THEN
                    )
                %ELSE
                    , value_size)
                %FI
        END
    %,



!
!       status.wlc.v = CDD$GET_STRING_LIST_ATT (context.rlu.r, entity.rlu.r,
!                       attribute.rlu.v, location.wlu.r ,
!                       [list_size.wwu.r]);
!

    $CDD$GET_STRING_LIST_ATT (context, entity, attribute, location,
        list_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$GET_STRING_LIST_ATT : FORTRAN;

            CDD$GET_STRING_LIST_ATT (%EXPAND $R(context), %EXPAND $R(entity),
                %EXPAND $R(attribute), %EXPAND $R(location)
                %IF %NULL (list_size) %THEN
                    )
                %ELSE
                    , list_size)
                %FI
        END
    %,


!
!     status.wlc.v = CDD$LOCK_NODE (context.rlu.r, [path.rt.dx] ,
!           [node.rlu.r] , location.wlu.r , [protocol.wt.dx] ,
!           [protocol-size.wwu.r] );
!

    $CDD$LOCK_NODE (context, path, node, location, protocol, protocol_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$LOCK_NODE           : FORTRAN;

            CDD$LOCK_NODE (%EXPAND $R(context)
                %IF %NULL(path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                        , 0, 0
                    %ELSE
                        , 0, node
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path, 0
                    %ELSE
                        , path, node
                    %FI
                %FI
                , %EXPAND $R(location)
                %IF %NULL (protocol) %THEN
                    %IF %NULL (protocol_size) %THEN
                        )
                    %ELSE
                        %ERROR ('Protocol-size cannot be used without protocol')
                        )
                    %FI
                %ELSE
                    , protocol
                    %IF %NULL (protocol_size) %THEN
                        )
                    %ELSE
                        , protocol_size)
                    %FI
                %FI
        END
    %,


!
!       status.wlc.v = CDD$NEXT_NODE (context.rlu.r, node.rlu.r, name.wt.dx,
!           [name-size.wwu.r] , [location.wlu.r] , [protocol.wt.dx] ,
!           [protocol-size.wwu.r]);
!

    $CDD$NEXT_NODE (context, node, name, name_size, location, protocol,
    protocol_size) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$NEXT_NODE           : FORTRAN;

            CDD$NEXT_NODE (%EXPAND $R(context), %EXPAND $R(node),
                %EXPAND $R(name)
                %IF %NULL (name_size) %THEN
                    %IF %NULL (location) %THEN
                        %IF %NULL (protocol) %THEN
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                %ERROR ('Protocol-size cannot be used without protocol')
                                )
                            %FI
                        %ELSE
                            , 0, 0, protocol
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                , protocol_size)
                            %FI
                        %FI
                    %ELSE
                        , 0, location
                        %IF %NULL (protocol) %THEN
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                %ERROR ('Protocol-size cannot be used without protocol')
                                )
                            %FI
                        %ELSE
                            , protocol
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                , protocol_size)
                            %FI
                        %FI
                    %FI
                %ELSE
                    , name_size
                    %IF %NULL (location) %THEN
                        %IF %NULL (protocol) %THEN
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                %ERROR ('Protocol-size cannot be used without protocol')
                                )
                            %FI
                        %ELSE
                            , 0, protocol
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                , protocol_size)
                            %FI
                        %FI
                    %ELSE
                        , location
                        %IF %NULL (protocol) %THEN
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                %ERROR ('Protocol-size cannot be used without protocol')
                                )
                            %FI
                        %ELSE
                            , protocol
                            %IF %NULL (protocol_size) %THEN
                                )
                            %ELSE
                                , protocol_size)
                            %FI
                        %FI
                    %FI
                %FI
        END
    %,


!
!       status.wlc.v = CDD$RENAME_NODE (context.rlu.r, [path.rt.dx] ,
!               [node.rlu.r] , name.rt.dx);
!

    $CDD$RENAME_NODE (context, path, node, name) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$RENAME_NODE         : FORTRAN;

            CDD$RENAME_NODE (%EXPAND $R(context)
                %IF %NULL(path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                    , 0, 0
                %ELSE
                    , 0, node
                %FI
            %ELSE
                %IF %NULL (node) %THEN
                    , path, 0
                %ELSE
                    , path, node
                %FI
            %FI
            , %EXPAND $R(name))
        END
    %,


!
!       status.wlc.v = CDD$RLSE_LOCKS (context.rlu.r, [path.rt.dx] ,
!             [node.rlu.r] , [options.rlu.v]);
!

    $CDD$RLSE_LOCKS (context, path, node, options) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$RLSE_LOCKS          : FORTRAN;

            CDD$RLSE_LOCKS (%EXPAND $R(context)
            %IF %NULL(path) %THEN
                %IF %NULL(node) %THEN
                    %IF %NULL(options) %THEN
                        )
                    %ELSE
                        , 0, 0, options)
                    %FI
                %ELSE
                    , 0, node
                    %IF %NULL(options) %THEN
                        )
                    %ELSE
                        , options)
                    %FI
                %FI
            %ELSE
                , path
                %IF %NULL(node) %THEN
                    %IF %NULL(options) %THEN
                        )
                    %ELSE
                        , 0, options)
                    %FI
                %ELSE
                    , node
                    %IF %NULL(options) %THEN
                        )
                    %ELSE
```

```
                        , options)
                    %FI
                %FI
            %FI
        END
    %,


!
!       status.wlc.v = CDD$SET_DEFAULT (context.rlu.r, [path.rt.dx] ,
!               [node.rlu.r]);
!

    $CDD$SET_DEFAULT (context, path, node) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$SET_DEFAULT         : FORTRAN;

            CDD$SET_DEFAULT (%EXPAND $R(context)
                %IF %NULL(path) %THEN
                    %IF %NULL (node) %THEN
                        %ERROR ('Either path or node must be specified')
                        , 0, 0)
                    %ELSE
                        , 0, node)
                    %FI
                %ELSE
                    %IF %NULL (node) %THEN
                        , path, 0)
                    %ELSE
                        , path, node)
                    %FI
                %FI
        END
    %,


!
!       status.wlc.v = CDD$SIGN_IN (context.wlu.r , [default_dir.rt.dx]);
!

    $CDD$SIGN_IN (context, default_dir) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$SIGN_IN             : FORTRAN;

            CDD$SIGN_IN (%EXPAND $R(context)
                %IF %NULL(default_dir) %THEN
                    )
                %ELSE
                    , default_dir)
                %FI
        END
    %,
!
!       status.wlc.v = CDD$SIGN_OUT (context.rlu.r);
!

    $CDD$SIGN_OUT (context) =
        BEGIN
            EXTERNAL ROUTINE
                CDD$SIGN_OUT            : FORTRAN;

            CDD$SIGN_OUT (%EXPAND $R(context))
        END
    %;


UNDECLARE
    %QUOTE $R;
```

0362 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

COM
REQ

SMGUSRTRM
LIS

SORLIB
REQ

SORTSHR
MAP

DEFS0
REQ

DKS
REQ

SORT32

SRTSPC
REQ

CDDTYPE
R32

SORTMERGE
MAP

SMGVECTOR
LIS

CHKPNT
REQ

CDDMAC
R32

SFKEYWRD
REQ

RECSYM
R32

SRTTRN
MAP

OPCODES
REQ